Name: *Student id:* *Section:* *Serial#.*

QUESTION ONE: Convert the following C++ code into equivalent assembly code. {8 pts}

```
int bb[12]={2,-1,5,-6,9,-7,-8,0,4,-3,-2,9}
int ctr = 0 ;
for(k=0; k<12; k++)
  {
    if(bb[k] >= 0)
      ctr++;
  }
cout<< "CTR =  " << ctr << endl;
```

```
include irvine32.inc                        ; Procedures Library

        .data
M9      BYTE    'CTR = ',0                  ; data definitions
BB      SDWORD  -2,-1,5,-6,9,-7,-8,0,4,-3,-2,-9

        .code
        MOV     ECX,lengthof BB             ; initializations
        MOV     ESI,OFFSET BB
        XOR     EAX,EAX

        LEA     EDX,M9                      ; cout << " CTR = ";
        CALL    WRITESTRING

L0:     CMP     DWORD PTR[ESI],0            ; if(aa[j]>0 then L3
        JL      L3

        INC     EAX                         ; EAX = CTR;

L3:     PUSHFD
        ADD     ESI,4                       ; inc j
        POPFD

        LOOP L0

        CALL    WRITEINT                    ; cout << CTR;
        CALL    CRLF                        ;cout<<endl;
```


this page isn't included.

Dr appx 5.

Name: *Student id:* *Section:* *Serial#:*

QUESTION TWO: Write a sequence of assembly instructions to perform each of the following tasks:

1) Set the even-numbered bits in ECX register to 1. Leave other bits in ECX unchanged {2 pt}

```
OR      ECX,55555555H
```

2) Divide the signed word values W1 / W2 {2 pt}

```
MOV     AX, W1
CWD
IDIV    W2
```

3) Give ONE instruction that sets ZERO flag to 1. {2 pt}

```
XOR     AX,BX
```

4) Set all bits in the flags register to 1. {2 pt}

```
PUSH    0FFFFFFFFH
POPFD
```

5) Shift the entire value in AX:BX ONE bit to the left. {2 pt}

```
SHL     BX, 1
RCL     AX, 1
```

6) Give ONE instruction to clear bit number 5 in AL register. Leave other bits in AL unchanged {2 pt}

```
AND     AL,0DFH
```

7) Store in EAX register the product of multiplying BH register by 1024; BH register may contain any unsigned value. (MUL and IMUL instructions are not allowed to be used). {2 pt}

```
MOVZX   EAX,BH
SHL     EAX,10
```

8) Swap the contents of 2 memory dwords (M1, M9) without using MOV & XCHG instructions. {2 pts}

```
PUSH    M1
PUSH    M9
POP     M1
POP     M9
```

Name: *Student id:* *Section:* *Serial*

QUESTION THREE: {8*1.5 = 12 pts}

a)
```
MOV   AX, 5C2FH
AND   AX, 0FCD7H
OR    AX, AX
```
AX = 5C 07 H

b)
```
MOV   AX, 4C9AH
MOV   BX, 8BH
SHLD  AX, BX, 4
```
AX = C9 A0 H

c)
```
MOV   AX, 6CB0H
MOV   CX, 4AC0H
IMUL  CL
```
AX = 14 00 H

d)
```
MOV   AX, 2309H
MOV   BX, 2670H
DIV   BL
```
AX = 09 50 H

e) What will be in registers AX and SP after executing the following instruction sequence?

```
      MOV   ESP, 7000H
      MOV   EBX, -1
      MOV   ECX, 3A74H
      PUSH  ECX
      CMP   CX, BX
      JL    L2
      PUSH  EAX
      AND   BX, 4C07H
L2:   XOR   BX, CX
```
BX = 76 73 H

SP = 6F F8 H

f) If "JL L2" is replaced by "JB L2", what will be in registers AX, SP after executing the above code?

BX = C5 8B H

SP = 6F FC H

Name: *Student id:* *Section:* *Serial#:*

QUESTION FOUR: {10 pts}

Write a complete assembly program that:

- Prompts the user to enter from the keyboard an integer value in the range1-20.
- Reads the value and validates it against the above given range limits. If an invalid value is entered, loop until a valid value is entered.
- Calls a procedure TRI8 that displays a triangle of stars "*" as shown in the example below
- The procedure accepts as a parameter the entered value in eax register.

For example, if you entered 5 from the keyboard, the output should be the following shape of stars:

```
*
**
***
****
*****
```

```
include irvine32.inc

        .data
M1      byte "Enter an integer value (0-20) please:",0
        .code
TRI8    proc

        mov     ecx, eax            ; ecx=#of lines to be displayed
        mov     ebx,1               ; ebx=#of "*" to be displayed
L21:    mov     edx,ebx             ; in line ecx
        mov     al, "*"

L22:    call    writechar           ; Display a line of stars
        dec     edx
        jnz     L22

        call    crlf
        inc     ebx                 ; Next line
        loop    L21
        ret
TRI8    endp

MAIN    proc
        mov     edx, offset M1      ; Display a message
L20:    call    writestring
        call    readint             ; read an integer in eax

        cmp     eax, 0              ; Validate entered value against
        jl      L20                 ; the given range (0-20)
        cmp     eax, 20
        jg      L20

        call    TRI8                ; Invoke procedure TRI8

        exit
MAIN    endp
        end     MAIN
```